JavaScript Output based test - 1

March-24/ JS/001 Time: 02:00hrs

JavaScript

Diploma in Advance Computing

March 2024

***What will be the output of the following code snippet?***

1. var x = 5;

console.log(x + "5");

1. var num = 10;

function increment() {

++num;

}

console.log (num);

increment();

1. var x = 10;

function fn() {

var x = 5;

console.log(x);

}

fn();

console.log(x);

1. var numbers = [1, 2, 3, 4, 5];

var newNumbers = numbers.slice(1, 4);

console.log(newNumbers);

1. var numbers = [1, 2, 3, 4, 5];

var sum = 0;

numbers.forEach(function (num) {

sum += num;

});

console.log(sum);

1. var numbers = [1, 2, 3, 4, 5];

var squaredNumbers = numbers.map(function (num) {

return num \* num;

});

console.log(squaredNumbers);

1. var numbers = [1, 2, 3, 4, 5];

var evenNumbers = numbers.filter(function (num) {

return num % 2 === 0;

});

console.log(evenNumbers);

1. function outer() {

var x = 10;

function inner() {

console.log(x);

}

return inner;

}

var closureFunction = outer();

closureFunction();

1. var x = 5;

function out() {

function inn() {

console.log(x);

}

var x = 10;

return inn;

}

var cFunction = out();

cFunction();

1. function functionDelay() {

for (var i = 1; i <= 5; i++) {

setTimeout(function () {

console.log(i);

}, 1000);

}

}

functionDelay();

1. var x=12;

var y=8;

var res=eval("x+y");

console.log(res);

1. function fn(x) {

    return (() => {

        console.log(x);

    })

}

fn(21);

1. function solve(arr, rotations){

if(rotations == 0) return arr;

for(let i = 0; i < rotations; i++) {

let element = arr.pop();

arr.unshift(element);

}

return arr;

}

console.log(solve([44, 1, 22, 111], 0));

1. function fn2(x) {

    return (() => {

        console.log(x);

    })()

}

fn2(21);

1. var a = 1;

var b = 0;

while (a <= 3)

{

a++;

b += a \* 2;

console.log(b);

}

1. var a = Math.max();

var b = Math.min();

console.log(a);

console.log(b);

1. var a = true + true + true \* 3;

console.log(a)

1. console.log(NaN === NaN);
2. console.log(typeof(NaN));
3. let sum = 0;

const a = [1, 2, 3];

a.forEach(Sum);

console.log(sum);

function Sum(element) {

sum += element;

}

1. const a = [1, 2, 3, 4, 5];

console.log(a.slice(2, 4));

1. console.log(parseInt("123Hello"));
2. var a = "hello";

var sum = 0;

for(var i = 0; i < a.length; i++) {

sum += (a[i] - 'a');

}

console.log(sum);

1. const fn = ( a, b, c ) => {

console.log(a, b, c);

};

fn(0, 1, 2);

1. var x = new Set ([4,5,6,7 ])

x.add(8);

console.log(x);

1. let a = 1

let b ='saleel'

let c = -1

function fn(...x) {

console.log(x)

}

fn(a, b, c);

1. var a = [34, 35, 45, 48, 49];

var b = [48, 55];

var x = [new Set([...a, ...b])];

console.log(x);

1. const x = new Set(['surat', 'saleel', 'baroda', '9850884228', 'surat']);

console.log(x.values());

1. const colors = ["red", "yellow", "blue"];

colors[2] = "purple";

console.log(colors);

1. const colors = ["red", "yellow", "blue"];

colors[5] = "purple";

colors.forEach((item, index) => {

console.log(index, item);

});

1. function fn1(book) {

book.title = "Redis";

book.cost = 1500;

book.total = book.cost + 1000;

};

const book = {

title: "Javascript",

author: "Saleel",

cost: 1200,

};

fn1(book);

console.log(book.title, book.cost, book.total);

1. var numbers = [1, 2, 3, 4, 5];

var sum = 0;

numbers.map((x) => sum+=x)

console.log(sum)

1. var num = 10;

function increment () {

num++;

}

console.log(num);

increment();

console.log(num);

1. console.log(parseInt("Hello123"));
2. let x = new Set([4, 5, 6, 7])

x.add(5);

x.add(8);

console.log(x);

1. console.log(typeof null);
2. let a = [1, 2, 3];

let b = a;

b.push(4);

console.log(a);

1. const PI = 3.14;
2. console.log(0.1 + 0.2 === 0.3);
3. function test() {

return;

console.log('Hello');

}

console.log(test());

1. console.log(typeof NaN);
2. let x = (function() {

return typeof arguments;

})();

1. let x = [1, 2, 3];

x[10] = 99;

console.log(x.length);

1. console.log([] + []);
2. console.log('5' - 3);
3. console.log('5' + 3);
4. let a = 1;

if (function f() {}) {

a += typeof f;

}

console.log(a);

1. console.log([] == ![]);
2. let x = 10;

(function() {

console.log(x);

var x = 20;

})();

1. for (var i = 0; i < 3; i++) {

setTimeout(() => console.log(i), 100);

}

1. const foo = () => {

return

{

bar: 1

};

};

console.log(typeof foo());

1. console.log([] + {});

console.log({} + []);

1. let a = (1, 5 - 1) \* 2;

console.log(a);

1. console.log(Boolean('false'));
2. const obj = { a: 1 };

Object.freeze(obj);

obj.a = 2;

console.log(obj.a);

56. console.log(typeof function() {});

57. console.log(1 + "2" + "2");

58. console.log("2" + 1);

console.log(2 + "1");

59. console.log("10" - "4" - "3" - 2 + "5");

60. console.log([] == []);

61. let x;

console.log(x + 1);

62. console.log(1 < 2 < 3);

console.log(3 > 2 > 1);

63. (() => {

let x = (y = 10);

})();

console.log(typeof y);

64. const arr = [1, 2, 3];

arr[10] = 5;

console.log(arr.length);

1. console.log("5" - 3);
2. let x = 3;

let y = "3";

console.log(x == y);

console.log(x === y);

1. let x = [1, 2, 3];

delete x[1];

console.log(x);

68. console.log(!!false);

69. console.log([] == 0);

70. console.log(false == '0');

71. const obj = { a: 1 };

const newObj = obj;

newObj.a = 2;

console.log(obj.a);

72. let x = (function() {

return typeof arguments;

})();

console.log(x);

73. console.log(!!undefined);

74. const a = [];

a[100] = 42;

console.log(a.length);

75. typeof typeof 1;

76. console.log(parseInt("123abc"));

77. console.log(parseInt("0xF"));

78. let a = [1, 2, 3];

a.length = 0;

console.log(a);

79. console.log(typeof []);

80. console.log(3 \*\* 2);

81. console.log(Number.isNaN("NaN"));

82. console.log('' == 0);

83. console.log(typeof Symbol("id"));

84. const x = {};

const y = { key: 'value' };

x[y] = 123;

console.log(x[y]);

85. const arr = [1, 2, 3];

arr.push(4);

console.log(arr);

86. What does Array.isArray([]) return?

87. const arr = [10, 20, 30];

console.log(arr.indexOf(20));

88. const arr = [1, 2, 3];

arr.pop();

console.log(arr);

89. const arr = ['a', 'b', 'c'];

console.log(arr.join('-'));

90. const arr = [1, 2, 3];

console.log(arr.length);

91. What does arr.shift() do in JavaScript?

92. let arr = [1, 2, 3];

arr.splice(1, 1);

console.log(arr);

93. const arr = [1, 2, 3];

const result = arr.map(x => x \* 2);

console.log(result);

94. const arr = [1, 2, 3];

console.log(arr.includes(2));

95. [1, 2, 3].map(x => x \* 2);

96. [1, 2, 3, 4].filter(x => x % 2 === 0);

97. [1, 2, 3, 4].reduce((a, b) => a + b);

98. [10, 20, 30].map((num, index) => num + index);

99. ['a', 'b', 'c'].map(x => x.toUpperCase());

100. [1, 2, 3, 4].filter(x => x > 2);

101. [5, 10].reduce((a, b) => a \* b, 2);

102. ['a', 'b', 'c'].reduce((a, b) => a + b);

103. [1, 2, 3].map(x => x \* x).filter(x => x > 4);

104. [2, 4, 6].filter(x => x > 3).reduce((a, b) => a + b);

105. function test() {

return;

}

console.log(test());

106. (function(a, b) {

return a + b;

})(2, 3);

107. var x = function() {

return 5;

};

console.log(typeof x);

108. let x = (function foo() {

return 10;

})();

console.log(typeof foo);

109. function add(x, y = 5) {

return x + y;

}

console.log(add(3));

110. function outer() {

let count = 0;

return function() {

count++;

return count;

};

}

const counter = outer();

console.log(counter());

console.log(counter());

111. function test() {

console.log(arguments.length);

}

test(1, 2, 3);

112. const arrow = () => ({ x: 1 });

console.log(arrow());

113. const obj = { a: 1, b: 2 };

console.log(Object.keys(obj));

114. const obj = { a: 1 };

obj.b = 2;

console.log(obj);

115. What is the result of typeof null?

116. const obj = { a: 1 };

console.log('a' in obj);

117. const a = { x: 1 };

const b = a;

b.x = 5;

console.log(a.x);

118. let user = {

name: "Alice",

greet() {

return `Hi, I'm ${this.name}`;

}

};

console.log(user.greet());

119. const obj = {};

console.log(obj.toString());

120. function outer() {

let count = 0;

return function() {

count++;

return count;

};

}

const fn = outer();

console.log(fn());

console.log(fn());

121. function makeAdder(x) {

return function(y) {

return x + y;

};

}

const add5 = makeAdder(5);

console.log(add5(3));

122. function counter() {

let val = 0;

return () => ++val;

}

let c = counter();

console.log(c(), c());

123. let funcs = [];

for (var i = 0; i < 3; i++) {

funcs.push(() => i);

}

console.log(funcs[0]());

124. function greet(name) {

return function() {

console.log(`Hello, ${name}`);

};

}

let greetJohn = greet("John");

greetJohn();

125. function createCounter() {

let count = 0;

return {

increment: () => ++count,

decrement: () => --count,

value: () => count

};

}

const counter = createCounter();

counter.increment();

counter.increment();

console.log(counter.value());

126. const arr = [1, 2, 3];

const [a, b] = arr;

console.log(a, b);

127. let x = `Sum is ${2 + 3}`;

console.log(x);

128. const sum = (a, b) => a + b;

console.log(sum(2, 3));

129. let obj = { a: 1, b: 2 };

let { a, b } = obj;

console.log(a, b);

130. const arr = [1, 2];

const newArr = [...arr, 3];

console.log(newArr);

131. let [a = 10, b = 20] = [5];

console.log(a, b);

**Answers**

**1. 55**

**2. 10**

**3. 5, 10**

4. **[2, 3, 4]**

5. **15**

6. **[1, 4, 9, 16, 25]**

7. **[2, 4]**

8. **10**

9. **10**

10. **6, 6, 6, 6, 6**

11. **20**

12. **No output**

13. **[44, 1, 22, 111]**

14. **21**

15. **4, 10, 18**

16. **-Infinity Infinity**

17. **5**

18. **false**

19. **number**

20. **6**

21. **[3, 4]**

22. **123**

23. **NaN**

24. **0, 1, 2**

25. **Set(5) {4, 5, 6, 7, 8 }**

26. **[1, 'saleel', -1]**

27. **[ Set(6) { 34, 35, 45, 48, 49, 55 } ]**

28. **{ 'surat', 'saleel', 'baroda', '9850884228' }**

29. **[ 'red', 'yellow', 'purple' ]**

30. **0 red**

**1 yellow**

**2 blue**

**5 purple**

**31. Redis 1500 2500**

**32. 15**

**33. 10, 11**

**34. NaN**

**35. Set(5) { 4, 5, 6, 7, 8 }**

**36. 'object'  
37. [1, 2, 3, 4]**

**38. const  
39. false**

**40. undefined**

**41. number  
42. object**

**43. 11**

**44. ""**

**45. 2**

**46. 53**

**47. 1function**

**48. true**

**49. undefined**

**50. 3 3 3**

**51. 'undefined'**

**52. '[object Object]' and 0**

**53. 4**

**54. true**

**55. 1**

**56. 'function'**

**57. "122"**

**58. "21" and "21"**

**59. "15"**

**60. false**

**61. NaN**

**62. true false**

**63. 'number'**

**64. 11**

**65.** 2

**66. true false**

**67. [1, undefined, 3]**

**68. false**

**69. true**

**70. true**

**71. 2**

**72. object**

**73. false**

**74. 101**

**75. "string"**

**76. 123**

**77. 15**

**78. []**

**79. "array"**

**80. 9**

**81. false**

**82. true**

**83. "symbol"**

**84. 123**

**85. [1, 2, 3, 4]**

**86. true**

**87. 1**

**88. [1, 2]**

**89. 'a-b-c'**

**90. 3**

**91. Removes the first element of the array**

**92. [1, 3]**

**93. [2, 4, 6]**

**94. true**

**95. [2, 4, 6]**

**96. [2, 4]**

**97. 10**

**98. [10, 21, 32]**

**99. ['A', 'B', 'C']**

**100. [3, 4]**

**101. 100**

**102. 'abc'**

**103. [4, 9]**

**104. 10**

**105.** undefined

**106. 5**

**107. "function"**

**108. "undefined"**

**109. 8**

**110. 1 2**

**111. 3**

**112. { x: 1 }**

**113. ['a', 'b']**

**114. { a: 1, b: 2 }**

**115. "object"**

**116. true**

**117. 5**

**118. Hi, I'm Alice**

**119. "[object Object]"**

**120. 1, 2**

**121. 8**

**122. 1, 2**

**123. 3**

**124. Hello, John**

**125. 2**

**126. 1 2**

**127. "Sum is 5"**

**128. 5**

**129. 1 2**

**130. [ 1, 2, 3 ]**

**131. 5 20**

**132.**